IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

A Proposed Approach to Federated Query Optimization

Kavitha Juliet[1], Rajeswari R P[2]

Dept. of CSE, RYMEC, Bellary, Karnataka[1,2]

Abstract: Recently, there is a growing need for query optimization that can effectively deal with federated database systems. Modern optimizers use a cost model to choose the best query execution plan (QEP) which heavily dependent on statistics maintained in the system catalog. Keeping such statistics up to date in the federation is troublesome due to local autonomy.This paper discuss query processing for a federated data base system. The main objective of the paper is to give general framework for query optimization in federated database system and enhancing the global query optimization.

Keywords: Federated Query Optimization, Query Execution Plan (QEP), Query Processing, Database System.

I. INTRODUCTION

The need for federated database services has increased dramatically in recent years. Within enterprises, IT infrastructures are often decentralized as a result of mergers, acquisitions, and specialized corporate applications, resulting in deployment of large federated databases.The main challenges in processing federated database queries originate from the data distribution, heterogeneity and autonomy[1]. The impact of data distribution has been well studied in distributed database research but not of heterogeneity and autonomy. It is therefore important to study how the problems arising from heterogeneity and autonomy can be appropriately handled in federated query processing.
Modern optimization techniques use statistics stored in system catalogue to estimate the cardinality of each intermediate step for all possible QEP and choosing the one with the lowest cost[6]. Outdated statistics causes misestimating intermediate cardinalities and hence choosing poor QEP. A poor choice of QEPs leads to unacceptable long query time and performance loss[2]. Keeping statistics up to date in the federation is a tedious task due to local autonomy principle which makes remote data sources statistics inaccessible to the federation[3][4]. Unfortunately remote data sources statistics maintained by federated server are the key for generating federated plans and keeping such statistics up to date is the base for choosing the best QEP[5]. This paper proposes federated query optimizer. Section 2 discusses the related work and section three gives the proposed federated query optimizer.

II. RELATED WORK

For federated queries, the federated LEO has estimation about the intermediate cardinalities of local and remote executed parts of such query. This estimation is based on the local statistics maintained by the optimizer. Supplementing intermediate actual cardinalities for remote parts is not easy task due to local autonomy principle which should be adhered in the federation. Several approaches, as in [8], developed to supplement this runtime information for remote parts of the query. These approaches can be categorized into two groups: immediate feedback and deferred feedback. Immediate feedback can be obtained by using (1) Proprietary monitoring tools built into remote data source runtime system which monitors query execution and supplement the federated server with the needed runtime information or (2) by cleverly rewriting SQL statements so that remote data source reply by query execution result as well as intermediate cardinalities. Rewriting can be done by representing each predicate by a common table expression (CTE) then the select query builds a union of the output from the CTE that represents the root operator and a count(*) statement with operator id and predicate id for each CTE or by utilizing a piped table user defined functions (UDFs), that does not modify the data and simply increments a counter for each row it pipes, inserted between every operator respectively predicate in the QEP to count the intermediate cardinalities. Deferred feedback can be obtained by generating additional statements that collect intermediate cardinalities at compilation time by using count (*) statements or using sampling techniques. Traditional query optimization techniques don't work well in the federation. In 1996, as in [8], a comprehensive query and processing-task trading negotiation framework is presented and work well in distributed database systems preserving black-box node autonomy .such framework considers the node that issue a request as a buyer for a product which locally decide its own optimal plan (minimum product price) and then a negotiation is done between the buyer and all remote nodes which have data (sellers) until the best offer is presented. Three principle forms of negotiation are defined bidding, auction and bargaining, [10] discusses the differences between them. After negotiation is done and the optimal seller is chosen processing of the remote query take place.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

III. PROPOSED FEDERATED FIVE PHASE QUERY OPTIMIZATION

In FDBS the detailed cost based query optimization is very difficult, if not possible. The federated query optimization is to find an execution plan for a user specified query that satisfies an optimal goal provided by the user. The two main factors that effect optimization cost is response time and total cost.
Response Time Optimization vs. Total Cost Optimization**:** Traditionally the optimization goal has been minimization of the total cost of execution, but in many applications, other factors such as response time, staleness of the data used in answering the query [12], or accuracy of the data [13] may also be critical. As has been pointed out previously [14, 15], optimizing for such an optimization goal requires the use of partial order dynamic programming technique. This technique is a generalization of the classical dynamic programming algorithm where the cost of each plan is computed as a vector and two costs are considered incomparable if neither is less than or equal to the other in all the dimensions2. It can be shown that if the cost is an l-dimensional vector, then the time and space complexity of the optimization process increases by a factor of 2 over classical dynamic programming
The objective of the optimization is to minimize the data shipping between sites.
Motivate by the previous work of [7] and [8], a framework has to be presented that combines these ideas to increase global query optimization capabilities. The proposed framework is shown in fig. 2.

A .Phase 1: Compilation Phase
For any data base request, the normal procedure involves:
1. Scanning which identifies language tokens.
2. Parsing-checking query syntax.
3. Validation-for all attributes and relation names.
4. An internal representation of the query is created.

B. Phase 2: Evaluating Best Execution (EBE) plan phase**:**

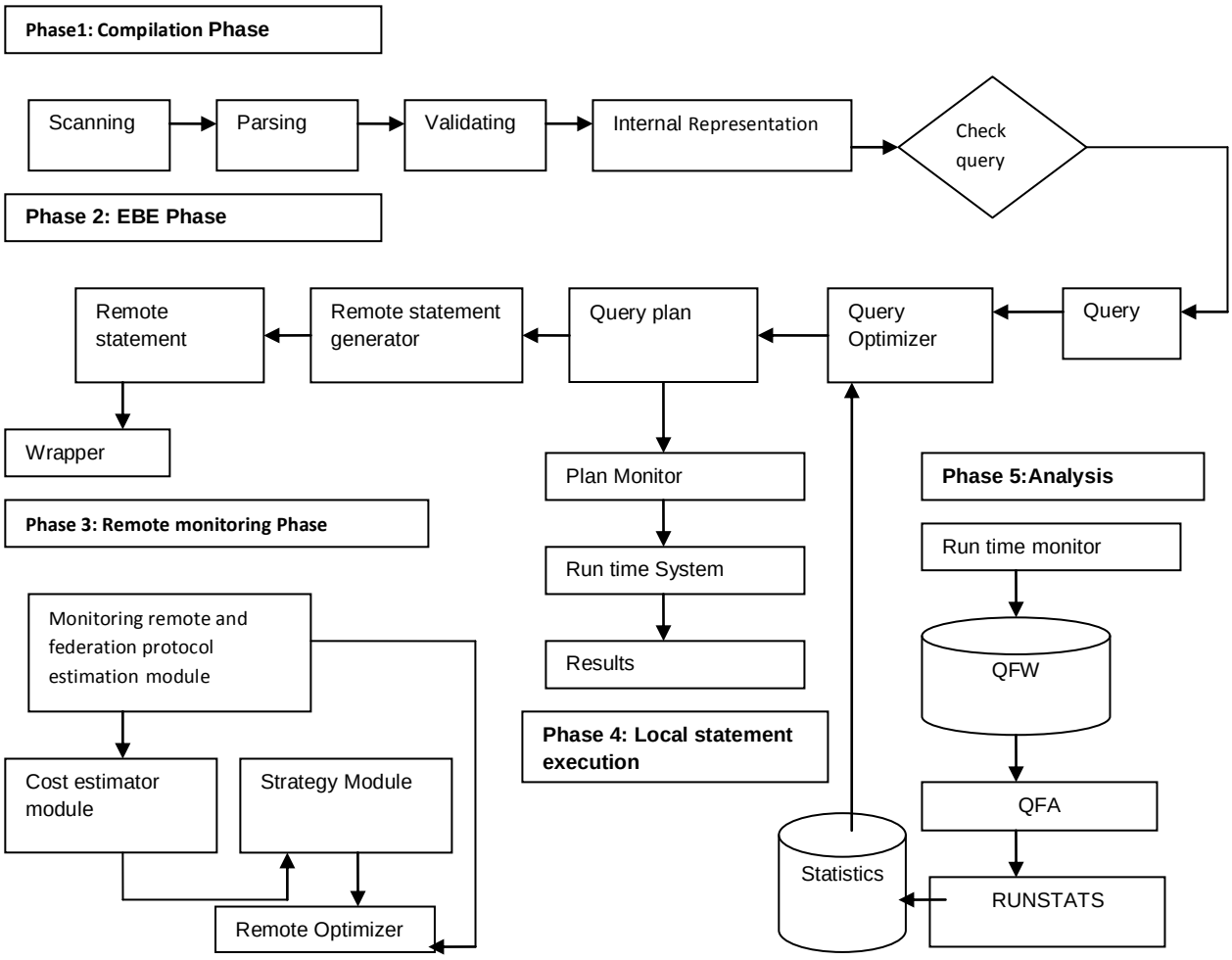

Figure 1: Proposed Five Phase Federated Query optimizer

For a query the best execution plan is determined by the federated optimizer depending on the statistics provided by local and remote data. This information is sent to plan monitor component (phase 4) which stores the cardinality estimate for such plan and supplement this to QFW of phase 5.
For remote parts of the query, remote statements generator generates remote statement and is send to wrapper to encapsulate what is unique for each remote data source and then it is send to phase3.

C. Phase 3: Remote Monitoring phase:
The aim of this phase is to collect the result of remotely executed queries. The steps are

1. Establishing the communication between federated optimizer and remote data source (phase 4).
2. The federated optimizer invite remote data source for executing the query.
3. The remote data source replies the best QEP for executing such query and give information to run time monitor component (Phase 4).

Recommendations can also be sent to federated server that will be helpful for federated server and may change its QEP for better optimization. Result is then transmitted to federated serverD. Phase 4: Local Statement Execution phase
For the overall QEP the plan monitor component stores the estimated cardinality in QFW coming from phase2. for local sub queries the federated server run time system execute local parts of QEP. The run time monitor stores the local cardinalities from run time system and remote cardinalities from phase 3. After this the remote and local sub queries should be communicated to give the result.

E. Phase 5 Analysis phase:
The query feedback analyzer (QFA) analyzes the information stored in query feedback warehouse by plan and run time monitors components and if error occurs updates the system catalog statistics.

IV.CONCLUSION

Distributed and Federated DBMS technologies have matured to the point where fairly sophisticated and reliable commercial systems are now available. Meanwhile, there are a number of issues that have yet to be satisfactorily resolved. This paper is dedicated to solve the problem of query execution speed by improving cardinality estimates for Federated DBMSs through better statistics. Applying this modification will give enhancement for federated query optimizer by (1) giving the federation the full autonomy as each remote data source will choose its own optimal QEP, (2) the statistics collection for federated queries will be precise as query answers are evaluated by the remote data source optimizer,(3) minimize the overhead of statistics collection with efficiency in processing federated queries and (4) enhancing the optimization process by permitting immediate feedback learning for remotely executed queries and hence permitting mid-query execution .Future work includes exposing the framework's functionality to mid-query re optimization with immediate feedback[11].

REFERENCES

[1] Sheth, A.P., and Larson, J.A.[1990] "federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases" ACM Computing Surveys,September 1990, pp.183-236.
[2] Tadeusz Morzy, Zbyszko Krolikowski, "Query Optimization in Multidatabase Systems: Solutions and Open Issues," dexa, p. 6, 10th International Workshop on Database & Expert Systems Applications, 1999.
[3] W.Litwin and A. ABDELLATIF. Multidatabase interoperability. IEEE Computer, December 986.
[4] L. M. Haas, E. T. Lin, M. A. Roth "Data integration through database federation "IBM SYSTEMS JOURNAL, VOL 41, NO 4, 2002.
[5] V. Markl, G. M. Lohman, V. Raman. LEO: An autonomic query optimizer for DB2. January 2003 IBM Systems Journal, Volume 42 Issue 1.
[6] A. Aboulnaga, P. Haas, S. Lightstone, G. Lohman, V. Markl, I. Popivanov, V. Raman. Automated Statistics Collection in DB2 Stinger. Proc. VLDB 2004.
[7] S. Ewen, M. Ortega-Binderberger, V. Markl."Learning Optimizer for a Federated Database Management System" BTW 2005: 87-106 [DBLP:conf/btw/EwenOM05].
[8] F.PENTARIS and Y.IOANNIDIS: Query Optimization in Distributed Networks of Autonomous Database Systems. ACM Transactions on Database Systems, Vol. 31, No. 2, June 2006.
[9] SU, S. Y., HUANG, C., HAMMER, J., HUANG, Y., LI, H., WANG, L., LIU, Y., PLUEMPITIWIRIYAWEJ, C., LEE, M., AND LAM, H. 2001.An internet-based negotiation server for e-commerce. VLDB
[10] N. Roussopoulos, "Materialized Views and Data Warehouses," SIGMOD Record 27, No. 1, 21–26, ACM, New York (1998).
[11] N. Kabra and D. DeWitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans," Proceedings of the ACM SIGMOD International Conference on Management of Data (June 1998), pp. 106–117.
[12] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In VLDB, 2000
[13] R. Avnur, J. M. Hellerstein, B. Lo, C. Olston, B. Raman, V. Raman, T. Roth, and K. Wylie. Control: Continuous output and navigation technology with refinement on-line. In SIGMOD, 1998.
[14] S. Ganguly, W. Hasan, and R. Krishnamurthy. Query optimizationfor parallel execution. In SIGMOD, 1992
[15] C. T. Yu, Z. M. Ozsoyoglu, and K. Lam. Optimization of distributed tree queries. JCSS, 1984.